Année 2024-2025

ING ESME Sudria
Projet Data Science
Build and deploy an open-source chatbot with Ollama
Construction et déploiement d'un chatbot open-source avec Ollama
Encadrant : Wajd MESKINI - wajd.meskini@gmail.com - 06 28 83 50 03
Keywords : NLP, Chatbot, Voicebot, DevOps
Language/Outils : Python, Ollama, vosk, pyttsx3

Introduction

L'objectif est de construire un assistant textuel (et par la suite vocal) pour aider des clients à avoir des réponses sur une FAQ.

- **Construction de l'environnement de travail** : Préparer l'environnement de code
- **Premières briques du chatbot** : Construire un chatbot simple avec des interactions de base (small-talk)
- **Utilisation du contexte** : Demander au chatbot de se baser sur le contexte pour répondre aux questions
- **Few-shot learning** : Rajouter des exemples de question réponse pour améliorer les performances du modèle
- **Text-to-Speech, Speech-to-Text** : Implémenter une interface web qui permet de tester l'enregistrement de voix, sa conversion en texte et sa lecture par un robot
- **Voicebot** : Connecter les deux modules à travers deux webservices

1 Construction de l'environnement de travail

Objectif : Préparer l'environnement de code

Assurez-vous de construire deux environnements conda. Le premier sera celui de Ollama (le chatbot textuel) et aura besoin de toutes les librairies nécessaires pour Ollama. Vous pourrez vous baser sur le Github de Ollama.

Vous aurez également à construire un second environnement qui se chargera de gérer la partie text-to-speech et speech-to-text, qui vous permettra de convertir votre chatbot en voicebot. Celui-ci - à priori - n'aura à contenir que vosk et pyttsx3. Libre à vous d'utiliser d'autres librairies de text-to-speech et speech-to-text si vous jugez celles-ci plus performantes.

1. Si vous utilisiez un fichier requirements.txt, comment ferez-vous pour installer toutes ces librairies en une seule ligne avec pip en partant du requirements.txt ?
2. Prendre connaissance des librairies installées. A quoi pourraient-elles servir ?
3. Assurez-vous d'avoir un micro fonctionnel pour cet exercice

2 Infrastructure

2.1 Librairies

Pour faire tourner un LLM sur Google Colab, nous allons commencer par installer les librairies nécessaires pour la bonne exécution du code. Nous allons figer les versions des librairies pour garantir que le code peut être réexécuter même si les librairies sont mises à jour. En effet, pip install cherche toujours la dernière version d'une librairie si une version n'est pas fixée, et celle-ci peut être incompatible avec votre code.

Nous allons commencer par installer les librairies nécessaires pour utiliser le LLM. Langchain et langchain-core contiennent des modules de base permettant de manipuler le LLM. Langchain-community contient Ollama, qui permet de télécharger et exécuter dans un serveur.

```
! pip install langchain==0.2.10 langchain-core==0.2.22
! pip install langchain-community==0.2.9
```

De plus, nous aurons besoin de xterm, qui permet d'exécuter un terminal à l'intérieur de Colab en parallèle du code. Nous l'utiliserons pour lancer notre LLM sur un serveur Ollama.

```
!pip install colab-xterm==0.2.0
```

2.2 Lancement du LLM

Nous allons d'abord initialiser xterm grâce à la commande suivante :

```
%load_ext colabxterm
```

Puis lancer un terminal xterm grâce à la commande suivante :

```
%xterm
```

Une fois que le terminal est lancé, nous allons installer ollama, télécharger le modèle que nous allons utiliser (phi3, de taille modeste pour facilier l'exécution) et le lancer.

```
curl -fsSL https://ollama.com/install.sh | sh
ollama serve & ollama pull phi3
ollama serve & ollama run phi3
```

Trois choses sont à noter à cette étape :

- Il faudra attendre que le modèle se télécharge et que le modèle se lance avant d'exécuter du code plus loin.
- D'autres modèles sont possibles. Phi3 n'est pas le meilleur, il est juste le meilleur rapport vitesse/performance. Se referer à cette liste si vous voulez tester d'autres modèles https ://ollama.com/library
- Les prompts de ce TP sont en anglais. Tous les LLM ne sont pas entraînés sur du texte francophone (ou même du texte d'autres langues que l'anglais). Si vous avez envie/besoin de tester des prompts en français, il faudra s'assurer de choisir un modèle capable de gérer un prompt en français (phi3 devrait le faire par défaut, mais mal)

Enfin, il faut noter que votre ordinateur ne peut pas accéder au service Colab. Il ne peut y accéder qu'à travers ngrok. Si vous voulez travailler sur Colab et bénéficier de sa GPU, il faudra trouver un moyen de le faire avec un compte gratuit ngrok. L'alternative serait de tout faire sur sa machine mais d'avoir un modèle lent. C'est un arbitrage à faire.

3 Premières briques du chatbot

Objectif : Construire un chatbot simple avec des interactions de base (small-talk)

3.1 Initialisation du modèle

Maintenant que le modèle est prêt à recevoir des inputs, il faudra initialiser une interface avec laquelle nous pourrons communiquer avec lui grâce à langchain. A ce stade, le paramètre MODEL prendra le nom du modèle qu'on a utilisé pour lancer notre serveur Ollama. Nous utiliserons phi3 dans ce TP mais d'autres modèles intéressants comme llama3 et qwen2 peuvent être utilisés.

```
from langchain_community.llms.ollama import Ollama
MODEL = "phi3"
model = Ollama(model=MODEL, temperature=0)
```

Nous pourrons tester de faire appel au modèle une première fois grâce à la commande suivante :

```
print(model.invoke("Give me an inspirational quote"))
```

3.2 Interface

Pour ne pas avoir à tester votre bot avec la ligne de commande, pensez à créer (ou reprendre) une interface web de chatbot afin de communiquer avec.

4 Utilisation du contexte

Objectif : Demander au chatbot de se baser sur le contexte pour répondre aux questions

A cette étape, cherchez sur le site web d'un service en ligne ou d'une compagnie quelconque leur FAQ. Idéalement ce serait une FAQ d'une taille modérée pour permettre de tester plusieurs façons de faire avec un temps de calcul acceptable.

Cette FAQ va servir de contexte : le modèle va répondre aux questions à partir de cette FAQ.

5 Few-shot learning

Objectif : Améliorer la performance du chatbot en lui donnant des exemples

En plus du contexte, il faudra rajouter des exemples pour améliorer sa performance. Documentez vous sur le few-shot learning pour voir sous quelle structure nous pouvons introduire ces exemples.

6 Text-to-Speech, Speech-to-Text

Objectif : Implémenter une interface web qui permet de tester l'enregistrement de voix, sa conversion en texte et sa lecture par un robot

6.1 Tester le speech-to-text et text-to-speech

Après installation des librairies correspondantes, il faudra commencer par lire le modèle de speech to text, et le tester sur un enregistrement de votre voix pour vous assurer qu'il permet bien de générer du texte.

Pareillement, vous devrez tester le modèle de text-to-speech et vous assurer qu'il peut bien lire du texte et le convertir en un fichier de voix.

6.2 Construction d'une interface

Pour déployer votre module de text-to-speech et speech-to-text, il faudra préparer les moyens de le tester. Ainsi, l'objectif sera de construire une interface web qui permet d'enregistrer de la voix, de l'envoyer en requête POST puis de lire un fichier de voix.

6.3 Transformation en webservice

L'objectif de cette étape revient à construire un webservice flask sur streamlit qui peut recevoir une requête POST avec un fichier de voix, la transformer en texte, puis faire le passage inverse en convertissant ce texte en voix et le renvoyant en réponse à la requête.

7 Voicebot

Objectif : Connecter les deux modules à travers deux webservices

L'étape finale est de lancer les webservices du chatbot et de la brique de text to speech/speech to text, et de les connecter par des requêtes : Ainsi, le serveur streamlit va récupérer le fichier de voix, le convertir en texte, envoyer le texte par requête POST au serveur Ollama, récupérer la réponse, la convertir en voix et la retourner à l'interface.

Le tout permet d'avoir un voicebot fonctionnel.

8 Sitographie

Modèle pré-entraîné francophone vosk (léger, 50 Mo)

https://alphacephei.com/vosk/models/vosk-model-small-fr-pguyot-0.3.zip

Modèle pré-entraîné francophone vosk (léger, 1.5 Go)

https://alphacephei.com/vosk/models/vosk-model-fr-0.6-linto-2.2.0.zip

Exemples d'usage de vosk

https://github.com/alphacep/vosk-api/tree/master/python/example